🦅

AWS EC2

🥗 접속

- ssh 접속 : ssh -i J8B307T.pem ubuntu@j8b307.p.ssafy.io
- 공인 ip : 54.180.157.223
- 도메인 : j8b307.p.ssafy.io

🍜 mysql 설치

- 접속 ip = 54.180.157.223
- id = root
- pwd = ssafy
- port : 3306

🥩 redis 설치

- port : 6379

🍔 redis sentinel

```
docker-compose up --scale redis-sentinel=5 -d
```

🌭 nginx 설치

- load balancing
- backend
- frontend

🚗 front - service

- docker deploy

```
docker run -d -p 3000:3000 --name front heesootory/pickpack-front
```

🍥 MSA docker network

- docker network 명 : pickpack

🍘 discovery-service

- port : 8761

```
docker run -d -p 8761:8761 --network pickpack --name discovery-service heesootory/pickpack-discovery
```

config server

- port : 7777

```
docker run -d -p 7777:7777 --network pickpack --name config-service heesootory/pickpack-config
```

🧁 apigateway-service

- port : 8000
- docker deploy

```
docker run -d -p 8000:8000 --network pickpack -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" --name a
```

- yml

```
server:
  port: 8000

eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka

spring:
  application:
    name: apigateway-service
  cloud:
    gateway:
      routes:
        - id: member-service
          uri: lb://MEMBER
          predicates:
            - Path=/api/member/**

        - id: chat-service
          uri: lb://CHAT
          predicates:
            - Path=/api/chat/**

        - id: item-service
          uri: lb://ITEM
          predicates:
            - Path=/api/item/**

        - id: flight-service
          uri: lb://FLIGHT
          predicates:
            - Path=/api/flight/**

logging:
  config: classpath:Logback-spring.xml
```

🍕 member-service

- 서비스 체크

```
http://54.180.157.223:8000/api/member/check
```

- docker deploy

```
docker run -d --network pickpack -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" --name member-service
```

- yml

```
server:
  port: 0
  tomcat:
    mbeanregistry:
      enabled: true

spring:
  application:
    name: member
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://54.180.157.223:3306/pickpack?
#    url: jdbc:mysql://localhost:3306/pickpack?
    username: root
    password:

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        show_sql: true

eureka:
  instance:
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
    lease-renewal-interval-in-seconds: 1
    lease-expiration-duration-in-seconds: 2
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka

management:
  endpoints:
    web:
      exposure:
        include: "*"
```

```
dependencies {
//    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation "com.querydsl:querydsl-jpa:${queryDslVersion}"
    annotationProcessor "com.querydsl:querydsl-apt:${queryDslVersion}"
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'io.micrometer:micrometer-registry-prometheus'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.h2database:h2'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}
```

chat-service

- 서비스 체크

```
http://54.180.157.223:8000/api/chat/check
```

- docker deploy

```
docker run -d --network pickpack -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" --name chat-service h
```

- yml

```
server:
  port: 0

spring:
  application:
    name: chat
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://43.201.19.30:3306/pickpack?
    #    url: jdbc:mysql://localhost:3306/pickpack?
    username: root
    password:

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
#        show_sql: true

eureka:
  instance:
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka
```

🍳 item-service

- 서비스 체크:

```
http://54.180.157.223:8000/api/item/check
```

- docker

```
docker run -d --network pickpack -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" --name item-service h
```

- yml

```
server:
  port: 8080
```

```
spring:
  application:
    name: item
  cloud:
    datasource:
      driver-class-name: com.mysql.cj.jdbc.Driver
      url: jdbc:mysql://localhost:3306/pickpack?
      username: root
      password:

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
logging:
  level:
    org.hibernate.sql : debug

eureka:
  instance:
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka
```

🍿 flight-service

- 서비스 체크 :

```
http://54.180.157.223:8000/api/flight/check
```

- docker

```
docker run -d --network pickpack -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" --name flight-service
```

- yml

```
server:
  port: 0
  tomcat:
    mbeanregistry:
      enabled: true

spring:
  application:
    name: flight
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://54.180.157.223:3306/pickpack?
    username: root # MySQL DB Username
    password: # MySQL DB password

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true

  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
```

```
logging:
  level:
    org.hibernate.sql : debug

eureka:
  instance:
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
    lease-renewal-interval-in-seconds: 1
    lease-expiration-duration-in-seconds: 2
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka

management:
  endpoints:
    web:
      exposure:
        include: "*"
```

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'io.micrometer:micrometer-registry-prometheus'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.h2database:h2'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'

    // Swagger 설정
    implementation group: 'io.springfox', name: 'springfox-swagger-ui', version: '3.0.0'
    implementation group: 'io.springfox', name: 'springfox-swagger2', version: '3.0.0'
    implementation group: 'io.springfox', name: 'springfox-boot-starter', version: '3.0.0'
}
```

하둡 설치

하둡 다운로드 및 압축 해제 (프로젝트 포함)

```
wget http://kdd.snu.ac.kr /~kddlab/Project.tar.gz
tar zxf Project.tar.gz
```

권한 설정 및 하둡 폴더 이동

```
sudo chown -R ubuntu:ubuntu Project
cd Project
sudo mv hadoop-3.2.2 /usr/local/hadoop
```

jdk 설치

```
sudo apt update
sudo apt install ssh openjdk-8-jdk ant -y
```

하둡 환경 셋 설정

```
./set_hadoop_env.sh
source ~/.bashrc
```

- core-site.xml 변경

```
cd ~/hadoop/etc/hadoop
vi core-site.xml
```

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/home/ubuntu/hadoop_tmp</value>
    </property>
</configuration>
```

ssh 키 생성

```
ssh-keygen -t rsa -P "" #저장할 파일 물어보면 default로 enter 치기
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

하둡 실행 준비

dfs 시작 및 확인

```
start-dfs.sh
jps
```

HDFS 디렉토리 생성

```
hdfs dfs -mkdir /user/input
```

HDFS 폴더 조회

```
hdfs dfs -ls /user
```

데이터를 HDFS에 넣기

```
hdfs dfs -put ~/Project/data /user/input
```

하둡 실행

하둡 파일 실행

```
hadoop jar pickpack.jar flightTicket /user/input /user/output
```

하둡 실행 결과 확인

```
hdfs dfs -ls /user/output
```

스파크 환경 구축

```
# spark 설치
sudo wget https://downloads.apache.org/spark/spark-3.2.3/spark-3.2.3-bin-hadoop3.2.tgz
sudo tar -xzvf spark-3.2.1-bin-hadoop3.2.tgz -C /usr/local

# Python 설치
sudo apt-get install -y python3-pip
# Python 버전 확인
python3 -V
# PySpark 설치
sudo pip3 install pyspark findspark

# Python library 설치
pip install tqdm
pip install pandas
pip install pymysql
pip install sqlalchemy
```